

Estructuras de Datos 1 - ST0245

Examen Parcial 1

Nombre

Departamento de Informática y Sistemas

Universidad EAFIT

Septiembre 4 de 2018

1 Recursión 20%

- (a) (10%) Considere el siguiente programa. ¿Cuál es la salida generada por $fun(11, 5)$? Como un ejemplo: $fun(10, 3) = 20$.

```
1  int fun(int n, int m){
2      if(n % m == 2) return n;
3      return fun(n + m, n - m);
4  }
```

- (i) 11
- (ii) 5
- (iii) 22
- (iv) 2
- (b) Considere el siguiente programa. ¿Cuál es la salida para $fun(1, 4)$? Como un ejemplo: $fun(1, 2) = 4$.

```
1  int fun(int m, int n){
2      if(m==0){
3          return (n+1);
4      }
5      if(m>0 && n==0){
6          return fun(m-1, 1);
7      }
8      int a=fun(m, n-1);
9      return fun(m-1, a);
10 }
```

- (i) 4
- (ii) 6
- (iii) 5
- (iv) 12

2 Complejidad 40%

- (a) (10%) ¿Es $O(2^{k+1}) = O(2^k)$?

- (i) Sí
- (ii) No

- (b) (10%) Considere el siguiente código escrito en Java. Encuentre la ecuación de recurrencia que mejor representa la complejidad asintótica en el peor de los casos.

```
1  int f(int n){
2      if(n <= 0){
3          return 1;
4      }
5      int a = f(n / 2);
6      int b = f(n / 2);
7      int res = 0;
8      for(int i = 0; i < n; i++){
9          res += (a*b);
10     }
11     return res;
12 }
```

- (i) $T(n) = 2T(n-1) + n$
- (ii) $T(n) = 2T(\frac{n}{2}) + n^2$
- (iii) $T(n) = 2T(\frac{n}{2}) + n$
- (iv) $T(n) = 2T(n-1) + (n-1)$

- (c) (10%) ¿Cuál de las siguientes afirmaciones es correcta para la función $func2(n, m)$?

```
1  void func2(int n, int m){
2      for(int i = 0; i < n; i++){
3          for(int j = 0; j < m; j++){
4              print(i, j);
5          }
6      }
7  }
```

- (i) Ejecuta $O(n + m)$ instrucciones
- (ii) Ejecuta $O(n.m)$ instrucciones
- (iii) Ejecuta $O(n^2 + m)$ instrucciones
- (iv) Ejecuta $O(m + n^2)$ instrucciones

- (d) (10%) Encuentre la ecuación de recurrencia que mejor representa la complejidad asintótica, para el peor de los casos, del siguiente código escrito en Java.

```

1  int f(int n){
2      if(n <= 0){
3          return 1;
4      }
5      int new_n = n / 5;
6      int re1 = f(new_n);
7      int re2 = f(new_n);
8      for(int i = 0; i < 4; ++i){
9          re1 = re1 + f(n / 7);
10     }
11     return re1 + re2;
12 }
```

- (i) $T(n) = 2 \times T(\frac{n}{5}) + 4 \times T(\frac{n}{7}) + C$
(ii) $T(n) = C \times T(\frac{n}{5}) + T(\frac{n}{7})$
(iii) $T(n) = C + 4 \times T(\frac{n}{5})$
(iv) $T(n) = C + 4 \times T(\frac{n}{7})$

3 Notación O 20%

- a (10%) Si $f(n) = O(n^2)$ y $g(n) = O(2^n)$, deduzca cuál es el valor de $O(f(n) \times g(n))$.

- (i) $O(2^n)$
(ii) $O(n^2)$
(iii) $O(2^n \times n^2)$
(iv) $O(2^n + n^2)$

- b (10%) Sea $T(n) = 0.003 \times \log_2 n + \log_2(\log_2 n)$. Calcule $O(T(n))$.

- (i) $O(\log_2(\log_2 n))$
(ii) $O(n^2)$
(iii) $O(\log_2 n)$
(iv) $O(n \log_2 n)$

4 ArrayList 20%

Nota: El método **add(n)** añade el elemento n en la última posición de la lista. El método **contains(n)** retorna verdadero si n está en la lista, sino retorna falso. El método **size()** retorna el tamaño de la lista. El método **get(i)** devuelve el elemento en la posición i de la lista.

- a (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de la siguiente función?

```

1  void fun1(ArrayList<Integer> list){
2      int a = -1, n=list.size();
3      for(int i=n; i >= 0; i--){
4          a = Math.max(a, list.get(i));
5      }
6      for(int i = 0; i < n; i++){
7          for(int j = 0; j < a; j++){
8              list.add(i*j);
9          }
10     }
11 }
```

- (i) $O(n^2)$
(ii) $O(\max(list) \times n)$
(iii) $O(\max(list) \times n^2)$
(iv) $O(n)$

Nota: La función $\max(list)$ es el elemento mayor de la lista.

- b (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, de insertar un elemento al principio de una lista hecha con arreglos (ArrayList)?

- (i) $O(n^2)$
(ii) $O(n)$
(iii) $O(\log n)$
(iv) $O(1)$